Le funzioni in c++

Classe 2C-2D, febbraio 2012

Un esempio.

Scriviamo un programma che calcoli il massimo di due numeri usando tre funzioni:

- la funzione **leggi** che legge un numero intero in input
- la funzione **fmax** che restituisce il massimo fra due numeri
- la funzione **scrivi** che stampa il risultato

Una funzione è una specie di "sottoprogramma" in cui vengono raccolte una serie di operazioni da eseguire. Ogni qualvolta si necessiti di eseguire questa serie di istruzioni (dal programma principale o da un'altra funzione), si richiamerà semplicemente il nome di codesta funzione.

Il codice del programma sopra proposto, sarà il seguente:

```
int fmax(int a, int b){
 int max;
 if(a>b){
    max=a;
 }
 else{
    max =b;
 };
 return max;
}

void leggi(int &a, int &b){
 printf("Inserisci un numero intero: ");
 scanf("%d",&a);
 printf("Inserisci un numero intero: ");
 scanf("%d",&b);
}

void scrivi(int a){
 printf("il massimo e' %d\n",a);
}

int main(){
 int x,y,vmax;

 leggi(x,y);
 vmax=fmax(x,y);
 scrivi(vmax);

 return 0;
}
```

Visibilità delle variabili

Innanzitutto va fatta subito una precisazione per quanto riguarda la visibilità delle variabili. Ogni variabile definita fra un'aperta e chiusa parentesi graffa, sarà definita e visibile **SOLO** all'interno di quelle parentesi graffe, all'interno di quel blocco di codice. Pertanto le variabili x,y e vmax definite nel programma principale (main), saranno visibili (si potranno usare) solo all'interno del programma principale. Stessa cosa dicasi della variabili max, a e b della funzione fmax: saranno visibili solo all'interno di tale funzione.

Si capisce però che una funzione dovrà solitamente operare su variabili presenti nel programma principale ed eventualmente restituire a questo un eventuale risultato; non potendo usare variabili in comune[1] dovrà esistere un modo per "passare" dal programma principale ad una funzione le variabili in questione. Vediamo come.

1 In realtà cioè è possibile anche se vivamente sconsigliato se non in casi particolari. Basterebbe semplicemente dichiarare le variabili al di fuori anche del programma principale, risultando quindi visibili a tutto il file. Tali variabili vengono dette variabili **globali**; per contrapposizione le altre sono dette variabili **locali.**

Ritorno di una variabile da una funzione

Concentriamoci sulla funzione fmax. Tale funzione dovrà ritornare al programma principale il valore corrispondente al massimo fra le due variabili a e b. La variabile che deve essere ritornata al programma principale è di tipo intero , e pertanto la funzione fmax è stata dichiarata di tipo intero (specificando **int** prima di fmax); quindi come ultima operazione della funzione viene richiamato il comando **return** seguito da un numero (o una variabile) intero da ritornare al programma principale. In questo modo il valore della variabile max della funzione fmax verrà assegnato alla variabile vmax del programma principale semplicemente con una chiamata del tipo:

```
vmax=fmax(x,y);
```

Chiaramente se vogliamo restituire al programma principale una variabile di tipo float, char o altro, dovremo dichiarare del tipo opportuno la funzione, la variabile interna alla funzione che restituiamo e la variabile del programma principale a cui viene assegnato questo valore.
Non sempre abbiamo però necessità di restituire qualcosa al programma principale. Si veda per esempio la funzione **scrivi**, dove semplicemente scriviamo dell'output ma senza restituire alcunché al programma principale. In questo caso la funzione verrà dichiarata di tipo **void** e non vi sarà alcuna chiamata al comando return.

Finora abbiamo visto come passare una variabile dalla funzione al programma principale. Ora vediamo come sia possibile il contrario.

Passaggio di parametri per ***valore***

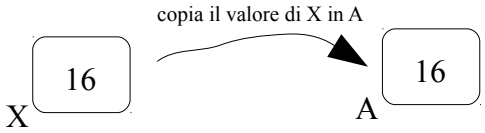
Ritorniamo sulla funzione fmax. Volendo calcolare il massimo di due numeri interi all'interno di questa funzione, dobbiamo conoscere il valore di tali valori. Tutti i parametri da passare alla funzione possono essere comunicati elencandoli fra parentesi dopo il nome della funzione. Nel programma principale, volendo calcolare il massimo fra le variabili x e y, chiameremo la funzione in questo modo:

```
vmax=fmax(x,y);
```

Chiaramente la funzione dovrà essere stata definita in modo opportuno, ovvero specificando che riceve in input 2 variabili di tipo intero.

```
int fmax(int a, int b){
```

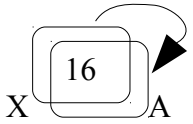
Come si può notare le variabili della funzione **a** e **b** sono distinte dalle variabile **x** e **y** del programma principale, sono due cose fisicamente diverse. Molto semplicemente il valore contenuto in **x** viene copiato nella variabile **a**, quello di **y** viene copiato nella variabile **b**.



Pertanto se verrà modificato il valore della variabile a all'interno della funzione, il valore della variabile x nel programma principale non verrà toccato.
Tale modalità di passare dei parametri ad una funzione viene detto **passaggio di parametri per valore**, proprio perché se ne copia solo il valore.

Passaggio di parametri per ***riferimento***

Il precedente funzionamento può essere cambiato se, nella dichiarazione della funzione, viene anteposta una **&** al nome delle variabili. In quel caso le variabili della funzione e della memoria corrisponderanno alla stessa identica celletta di memoria.



In questo modo qualsiasi modifica fatta alla variabile **a** all'interno della funzione, sarà automaticamente apportato anche alla variabile **x** essendo quest'ultima la stessa identica cosa di **a**.
Questo tipo di passaggio di parametri ad una funzione è detto **passaggio di parametri per**

riferimento, poiché non si passa semplicemente il valore dei parametri ma il riferimento all'indirizzo in memoria della variabile x.

Nell'esempio in questione con la chiamata della funzione **leggi** nel programma principale

```
leggi(x,y);
```

si vuole impostare il valore delle variabili x ed y direttamente all'interno della funzione leggi. Pertanto bisognerà usare il passaggio di parametri per riferimento nella definizione di tale funzione, ovvero anteponendo una **&** prima del nome dei parametri:

```
void leggi(int &a, int &b){
```

Il valore assegnato ad **a** e **b** in tale funzione, verrà quindi assunto dalle variabili **x** ed **y** del programma principale.

Passaggio di array

Ad una funzione può essere passata non solo una variabile standard (int, float, char...) ma anche un vettore. Ecco l'esempio di una funzione che ritorna l'elemento massimo di un vettore di interi:

```
int fmaxv(int v[10]){
     int max;
     max=v[0];
     for(int i=1; i<10; i++){
        if(v[i]>max) max=v[i];
     }
     return max;
}

int main(){
    int m,vettore[10];
    m=fmaxv(vettore);
    printf("il massimo è %d\n",m);

    return 0;
}
```

Si noti che un vettore viene **sempre** passato per **riferimento senza** specificare la **&**. I vettori possono solo essere passati in questo modo e non è possibile passarli per valore.

Esercizi

- Riscrivi il codice per il calcolo del massimo fra due numeri usando le tre funzioni come da esempio.
- Scrivi la funzione per il calcolo dell'elemento massimo di un vettore.

```
void massimo(float v[10])
```

- Scrivi una funzione che ordini un vettore.

```
void ordina(float v[10])
```

- Riprendi qualcuno dei programmi già fatti e riscrivilo usando le funzioni (quello del campo fiorito, i vari esercizi fatti sull'ordinamento, sui vettori etc).
- Scrivi una funzione che calcoli il fattoriale di un numero dato come parametro e che ne restituisca il valore al programma principale.

```
float fattoriale(int n)
```

- Scrivi una funzione che calcoli l'elevamento a potenza (intera) di un numero reale, e che ne restituisca in uscita il valore al programma principale.

```
float potenza(float x, int n)
```

- Scrivi una funzione che calcoli il minimo comune multiplo fra due numeri, restituendolo in uscita

```
float mcm(int x, int y)
```

- Scrivi una funzione che calcoli il massimo comun divisore fra due numeri, restituendolo in uscita

```
float MCD(int x, int y)
```

- Scrivi una funzione che calcoli la media di due numeri reali

```
float media(float x, float y)
```

- Scrivi una funzione che calcoli la media degli elementi di un vettore di 100 elementi

```
float media(float v[100])
```

- Scrivi una funzione che riempia un vettore con numeri acquisiti da tastiera.

```
void leggiv(float v[10])
```

- Scrivi una funzione che riempia un vettore con numeri reali random compresi tra 3 e 10.

```
void setrandom(float v[10])
```

- Scrivi una funzione che stampi a video un vettore.

```
void stampav(float v[10])
```

- Scrivi una funzione che copi un vettore v dentro un altro vettore w.

```
void copiav(float v[10], float w[10])
```

- Scrivi una funzione che scambi il valore di due variabili **a** e **b**.

```
void scambia(int &a, int &b)
```

- Scrivi un programma che calcoli area e perimetro di varie figure geometriche, facendo scegliere all'utente cosa calcolare. Suddividere il programma in differenti funzioni (es: quella per il calcolo delle proprietà di un triangolo, di un trapezio, di un rombo... quella per il calcolo del perimetro, piuttosto che dell'area.. etc)
- Scrivi la funzione **leggim** che legga da tastiera un numero intero maggiore di 0. Se il numero è maggiore o uguale a 0, la funzione stessa dovrà prendersi cura di stampare un messaggio d'errore e rileggere un'altra volta il numero.

```
void leggim(int &a)
```